CHINA OpenInfra Days

katacontainers

CHINA OpenInfra Days

IT大咖说 知识共享平台

Bring Security to the Container World, and More

Kata Containers 1.0 and Beyond

王旭 hyper.sh, Kata Containers Arch Committee
张伟 华为, Kata Containers Arch Committee

*"All problems in computer science can be solved by another level of indirection, except of course for the problem of **too many indirections**."*

----David Wheeler

计算机科学界的所有问题都可以通过再增加一个间接层来解决，当然，除了间接层太多这个问题。

Agenda

- Virtualization or Containerization: it's a Question
- From runV to Kata Containers
- Kata Containers vs. gVisor: More than Secure
- Kata for Kubernetes: secure, multi-tenancy, devices, etc.
- Let's Get Start

What's Kata Containers

- A container runtime, like runC
- Built w/ virtualization tech, like VM
- Initiated by hyper.sh and Intel®
- Hosted by OpenStack Foundation
- Contributed by Huawei, Google, MSFT, etc.

Kata Containers is Virtualized Container

The Evolution of Containers

- Processes in time-sharing operating systems
- Process groups: related processes `ps aux|less`
- Chroot: filesystem restriction
- Namespaces and cgroups: more isolation
- Docker and Docker images: make OS commodity
- OCI and CRI: standardization

app centric | OS feature | package | runtime

The Evolution of Virtualization

- Partition in mainframes and small computers
- Virtualization by instruction translations
- Para-Virtualization
- Hardware-assisted virtualization
- PV drivers, SR-IOV, and more.

system view | OS supervisor | isolation | hardware aware*

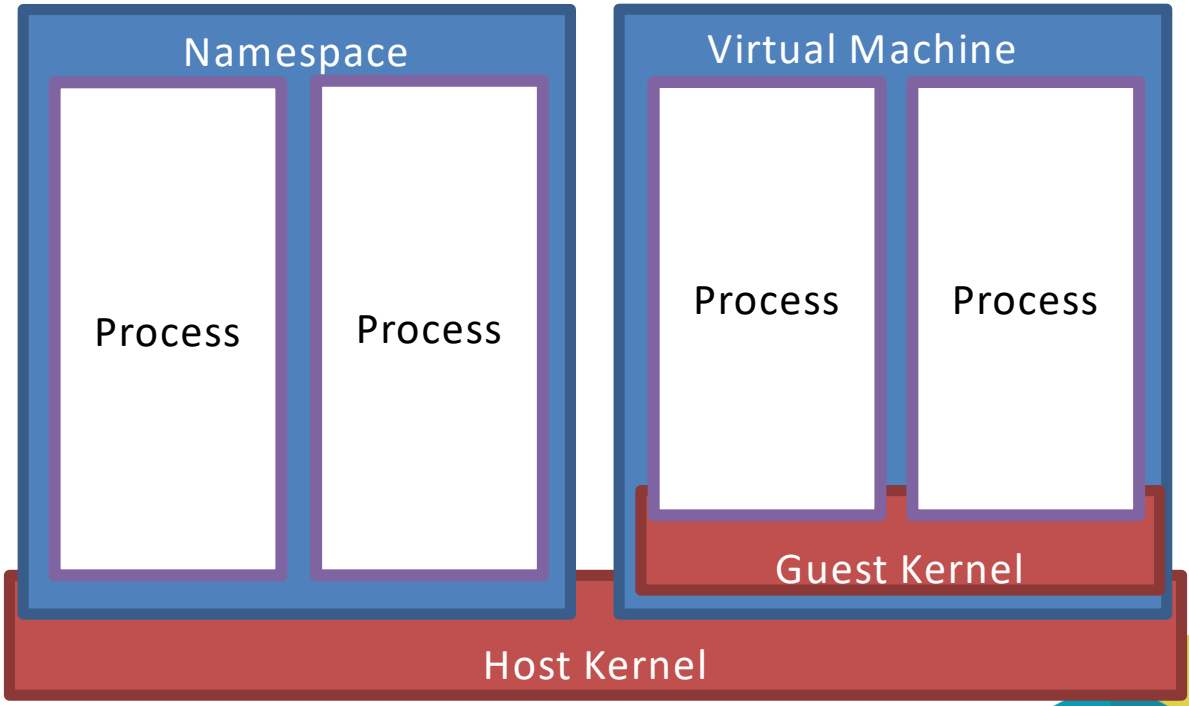
The Difference

	Containerization	Virtualization
	app centric, OS feature, package, runtime	system view, OS supervisor, isolation, hardware aware*
View Point	From application to OS; The environment of an app	From hardware to OS; intercept the instructions of guest system
Design Goal	Kernel ABI, including syscall interfaces, procfs/sysfs, etc.	A machine like the host, could run a kernel on top of it
Implementation	Inside kernel, a feature of the scheduler	A supervisor of the kernel, a scheduler of the schedulers

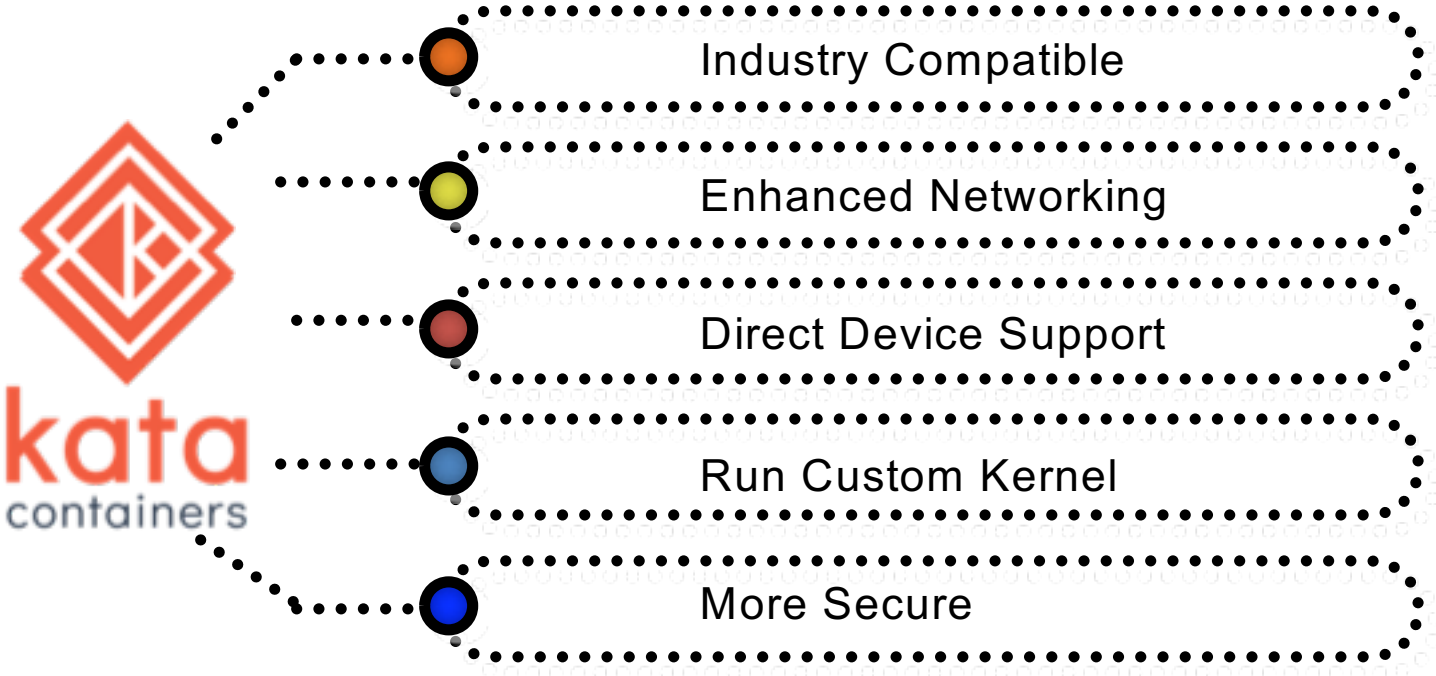
And the Similarity

- Run processes on kernel ABI
  - Looks like a whole OS from the process view
- Kind of boundary
  - Isolate processes from outside

VM is a type of container indeed



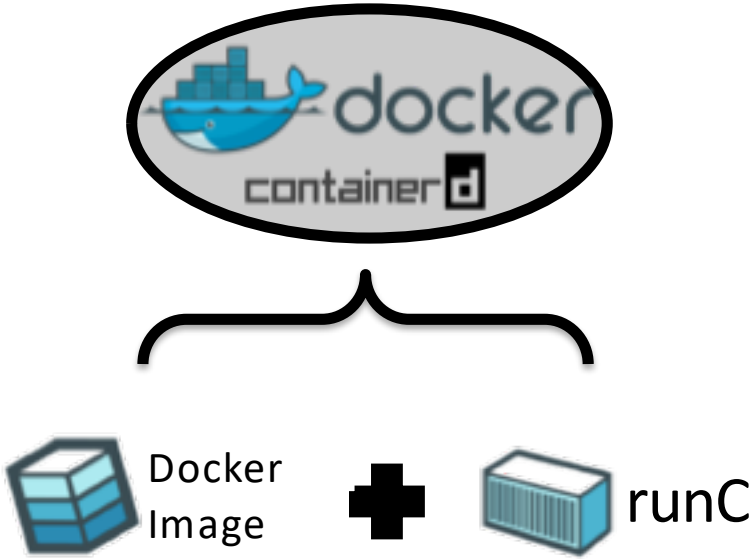
Combine the Best from Both V* and C*



Make VM run like containers

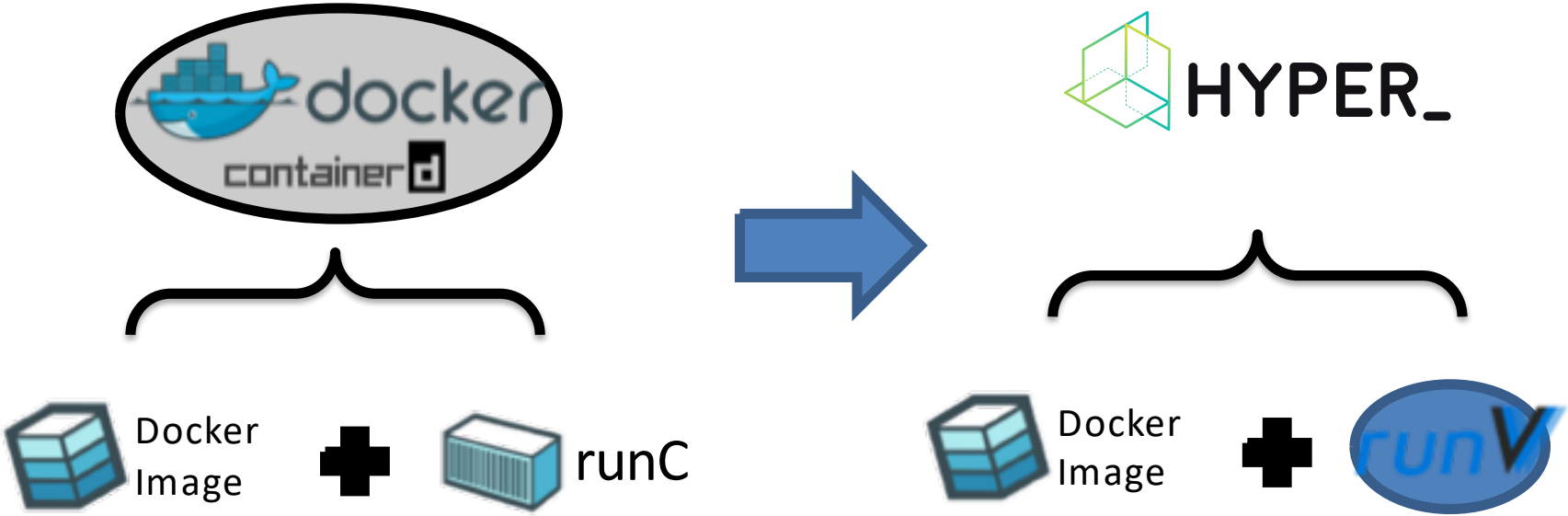
FROM RUNV TO KATA CONTAINERS

Containers: the Pros and Cons



- Pros
  - 👍 Fast: sub-second boot
  - 👍 Light: MBs of a Docker image and Git-like DevOps
  - 👍 Portable
- Cons
  - 👎 Weak isolation, shared kernel
  - 👎 It is NOT a "Machine"
    - 👎 No passthrough or other accelerating features for VMs

Make VM run like Container



From runV to Kata Containers

Intel®
Clear Containers

HYPER.SH
runV

kata containers
Announced project

Release 1.0

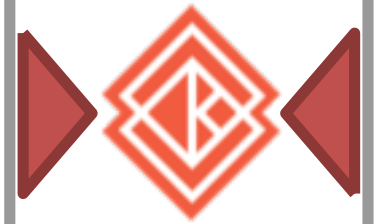
May 2015

Dec 2017

May 2018

Kata is Virtualized Container

Multi Architecture
Multi Hypervisor
Full Hotplug
K8s Multi Tenancy
VM templating
Frakti native support
Traffic Controller net



Direct Device Assignment
SRIOV
NVDIMM
Multi-OS
KSM throttling
CRI-O native support
MacVTap, multi-queue net

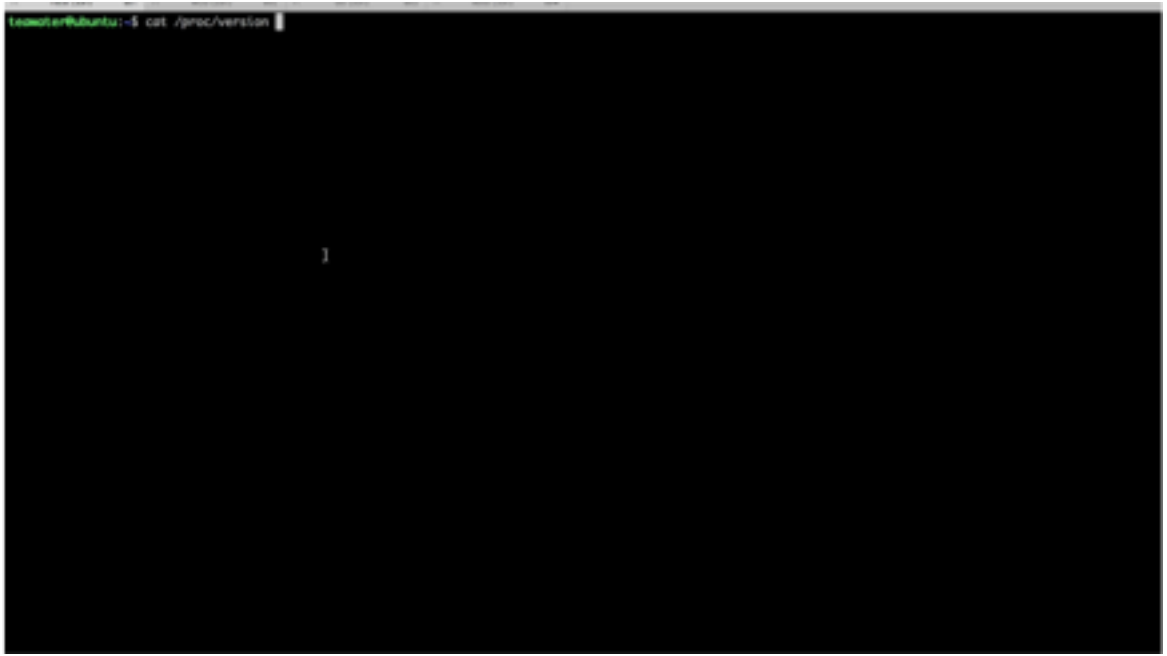
Kata Containers inherits the features from both runV and Clear Containers

Fast as Container

```
[root@localhost ~]# hyperctl pull ubuntu:latest
...
[root@localhost ~]# hyperctl run -t ubuntu
root@ubuntu-2994825143:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@ubuntu-2994825143:/# exit
exit
[root@localhost ~]# hyperctl run -d ubuntu
POD id is pod-aEafFYramp
Time to run a POD is 143 ms
[root@localhost ~]# hyperctl list
POD ID              POD Name            VM name             Status
pod-CMLStRNiKG      ubuntu-2994825143                       succeeded
pod-aEafFYramp      ubuntu-3972307775   vm-TeLKCtGBcF       running
```

Secure as VM (1)

- Namespace based Containers share the kernel with host
- More than 400 Kernel CVEs in 2007, similar to before
- With some of them, you may escape to host with root privilege
- Such as CVE-2017-5123
- Here is a simple demo



Video: Try CVE-2017-5123 in runC

Secure as VM (2)

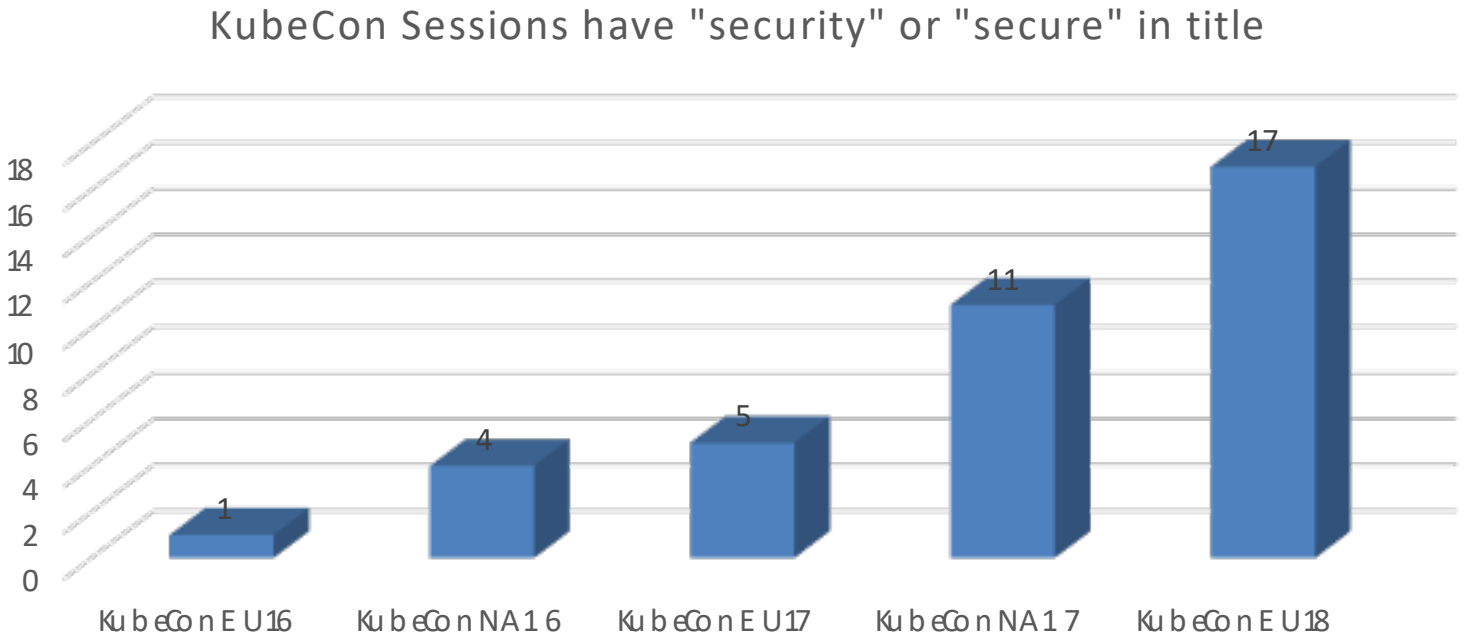
- With Kata, the App is isolated with VM
- Even with an exploit in guest kernel, the attacker may not escape to the host
- An CVE like CVE-2017-5123 will not take effect in kata containers
- Here is the second part of the demo

Video: Try CVE-2017-5123 in Kata

Kata Containers vs. gVisor

SECURITY AND MORE BENEFITS

Time: Security Become Popular



Comparison (1): General

	gVisor(ptrace)	gVisor(kvm)	runc	runv	aliuk
memcached	11.8M/s	13.5M/s	66.5M/s	57.8M/s	82.7M/s

Memcached net_rate (bigger is better)

	gVisor(kvm)	runc	aliuk
mysql	22773.19ms	1371.44ms	1673.86ms

mysql oltp.lua latency (smaller is better)

	runC	runV	gVisor	AliUK	Notes
self contained	NA	Completed	Close to Completed	Completed	Some of the syscall in gVisor depends on host kernel
performance	Very High	High	Low	Very High	
overhead	Very Low	Low	Low	Low	
Compatibility	NA	Very Strong	Strong	Weak	Part of Linux ABI has not supported in gVisor
Security	Weak	Very Strong	Strong	Very Strong	Attack surface of gVisor is a bit bigger than others

Source of the benchmark in this page: Alibaba (in Chinese)

Comparasion (2): Networking

[4] 169.00-170.00 sec	61.9 MBytes	520 Mbits/sec	0	0.00 Bytes
[4] 170.00-171.00 sec	89.5 MBytes	750 Mbits/sec	0	0.00 Bytes
[4] 171.00-172.00 sec	119 MBytes	1.00 Gbits/sec	0	0.00 Bytes
[4] 172.00-173.00 sec	81.0 MBytes	679 Mbits/sec	0	0.00 Bytes
[4] 173.00-174.00 sec	84.0 MBytes	704 Mbits/sec	0	0.00 Bytes
[4] 174.00-175.00 sec	110 MBytes	918 Mbits/sec	0	0.00 Bytes
[4] 175.00-176.00 sec	153 MBytes	1.28 Gbits/sec	0	0.00 Bytes
[4] 176.00-177.00 sec	109 MBytes	913 Mbits/sec	0	0.00 Bytes
[4] 177.00-178.00 sec	83.6 MBytes	701 Mbits/sec	0	0.00 Bytes
[4] 178.00-179.00 sec	76.9 MBytes	645 Mbits/sec	0	0.00 Bytes
[4] 179.00-180.00 sec	119 MBytes	1.00 Gbits/sec	0	0.00 Bytes
[4] 180.00-181.00 sec	111 MBytes	927 Mbits/sec	0	0.00 Bytes
[4] 181.00-182.00 sec	128 MBytes	1.08 Gbits/sec	0	0.00 Bytes
[4] 182.00-183.00 sec	66.0 MBytes	554 Mbits/sec	0	0.00 Bytes
[4] 183.00-184.00 sec	90.8 MBytes	762 Mbits/sec	0	0.00 Bytes
[4] 184.00-185.00 sec	152 MBytes	1.28 Gbits/sec	0	0.00 Bytes
[4] 185.00-186.00 sec	153 MBytes	1.28 Gbits/sec	0	0.00 Bytes
[4] 186.00-187.00 sec	41.9 MBytes	353 Mbits/sec	0	0.00 Bytes
[4] 187.00-188.00 sec	67.2 MBytes	564 Mbits/sec	0	0.00 Bytes
[4] 188.00-189.00 sec	78.3 MBytes	657 Mbits/sec	0	0.00 Bytes
[4] 189.00-190.00 sec	61.8 MBytes	518 Mbits/sec	0	0.00 Bytes
[4] 190.00-191.00 sec	87.8 MBytes	737 Mbits/sec	0	0.00 Bytes
[4] 191.00-192.00 sec	89.1 MBytes	747 Mbits/sec	0	0.00 Bytes
[4] 192.00-193.00 sec	146 MBytes	1.23 Gbits/sec	0	0.00 Bytes
[4] 193.00-194.00 sec	81.3 MBytes	682 Mbits/sec	0	0.00 Bytes
[4] 194.00-195.00 sec	104 MBytes	875 Mbits/sec	0	0.00 Bytes
[4] 195.00-196.00 sec	66.6 MBytes	559 Mbits/sec	0	0.00 Bytes
[4] 196.00-197.00 sec	77.2 MBytes	647 Mbits/sec	0	0.00 Bytes
[4] 197.00-198.00 sec	125 MBytes	1.05 Gbits/sec	0	0.00 Bytes
[4] 198.00-199.00 sec	117 MBytes	980 Mbits/sec	0	0.00 Bytes

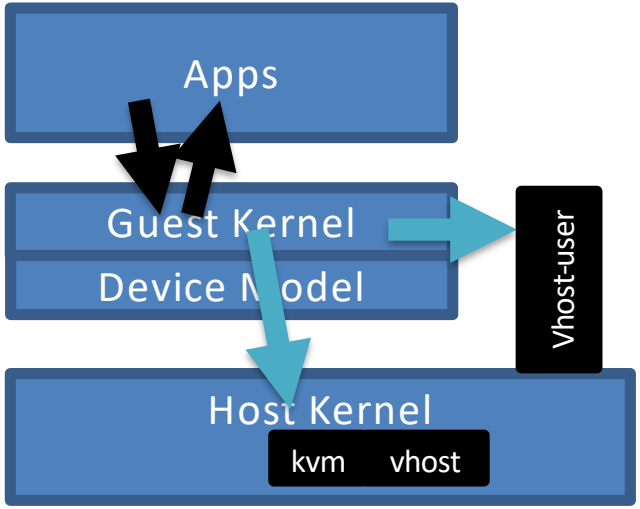
gVisor

[4] 170.00-171.00 sec	6.12 GBytes	52.6 Gbits/sec	0	625 KBytes
[4] 171.00-172.00 sec	6.15 GBytes	52.8 Gbits/sec	0	625 KBytes
[4] 172.00-173.00 sec	6.28 GBytes	53.9 Gbits/sec	0	625 KBytes
[4] 173.00-174.00 sec	6.47 GBytes	55.6 Gbits/sec	0	625 KBytes
[4] 174.00-175.00 sec	6.32 GBytes	54.3 Gbits/sec	0	625 KBytes
[4] 175.00-176.00 sec	6.15 GBytes	52.8 Gbits/sec	0	625 KBytes
[4] 176.00-177.00 sec	6.44 GBytes	55.3 Gbits/sec	0	625 KBytes
[4] 177.00-178.00 sec	6.12 GBytes	52.6 Gbits/sec	0	625 KBytes
[4] 178.00-179.00 sec	6.27 GBytes	53.9 Gbits/sec	0	625 KBytes
[4] 179.00-180.00 sec	6.30 GBytes	54.1 Gbits/sec	0	625 KBytes
[4] 180.00-181.00 sec	6.42 GBytes	55.1 Gbits/sec	0	625 KBytes
[4] 181.00-182.00 sec	6.24 GBytes	53.6 Gbits/sec	0	625 KBytes
[4] 182.00-183.00 sec	6.20 GBytes	53.3 Gbits/sec	0	625 KBytes
[4] 183.00-184.00 sec	6.30 GBytes	54.2 Gbits/sec	0	625 KBytes
[4] 184.00-185.00 sec	6.28 GBytes	54.0 Gbits/sec	0	625 KBytes
[4] 185.00-186.00 sec	6.12 GBytes	52.6 Gbits/sec	0	625 KBytes
[4] 186.00-187.00 sec	6.37 GBytes	54.7 Gbits/sec	0	625 KBytes
[4] 187.00-188.00 sec	6.37 GBytes	54.7 Gbits/sec	0	625 KBytes
[4] 188.00-189.00 sec	6.45 GBytes	55.4 Gbits/sec	0	625 KBytes
[4] 189.00-190.00 sec	6.26 GBytes	53.8 Gbits/sec	0	625 KBytes
[4] 190.00-191.00 sec	6.34 GBytes	54.5 Gbits/sec	0	625 KBytes
[4] 191.00-192.00 sec	6.34 GBytes	54.4 Gbits/sec	0	625 KBytes
[4] 192.00-193.00 sec	6.29 GBytes	54.0 Gbits/sec	0	625 KBytes
[4] 193.00-194.00 sec	6.23 GBytes	53.5 Gbits/sec	0	625 KBytes
[4] 194.00-195.00 sec	6.15 GBytes	52.8 Gbits/sec	0	625 KBytes
[4] 195.00-196.00 sec	6.10 GBytes	52.4 Gbits/sec	0	625 KBytes
[4] 196.00-197.00 sec	6.31 GBytes	54.2 Gbits/sec	0	625 KBytes
[4] 197.00-198.00 sec	6.32 GBytes	54.3 Gbits/sec	0	625 KBytes
[4] 198.00-199.00 sec	6.14 GBytes	52.7 Gbits/sec	0	625 KBytes
[4] 199.00-200.00 sec	6.12 GBytes	52.6 Gbits/sec	0	625 KBytes

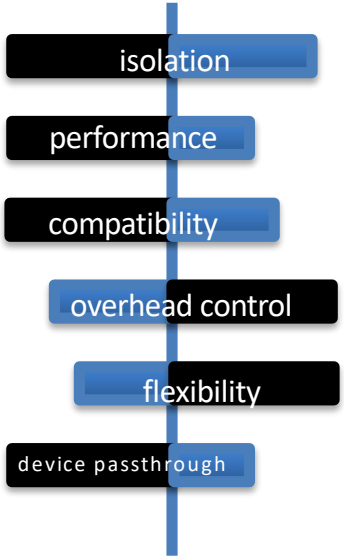
Host

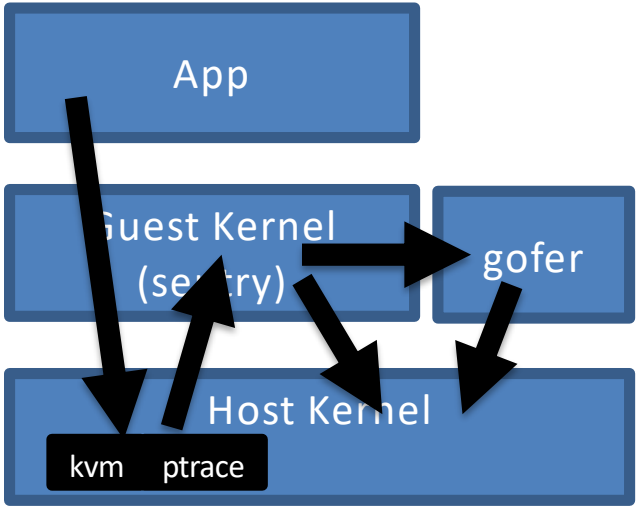
Data source: CloudNetEngine

Architecture



Kata Containers





gVisor

Community

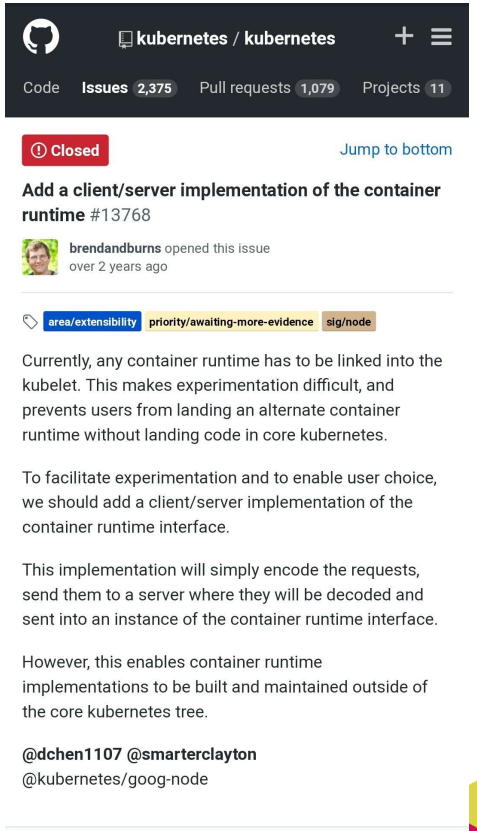
- Kata Containers
  - A Community Project hosted by OpenStack Foundation
    - Contributors: Hyper.sh, Intel, Huawei, ARM, Dell-EMC, Google, etc.
  - Based on and contribute to communities including kernel, qemu, kubernetes
  - OCI Partner Member, and Compatible with OCI
  - Kubernetes Support: CRI (frakti) & OCI (CRI-O and Containerd)
- gVisor
  - A Google Project
  - A new “kernel” written in Go
  - OCI: Compatible
  - Kubernetes: via CRI-O and Containerd, as an OCI client
- And More…

Hard Multi-Tenancy & Heterogeneous Workload Supports

KATA CONTAINERS AND CRI

A Brief History of K8S CRI

- Once upon a time…
  - rkt was added into kubelet as the 2nd runtime.
    - Increased the complexity on maintenance
  - Docker (the 1st runtime) introduced more and more feature.
    - Don’ t like a simple runtime any more
  - Hyper.sh joined the community and tried to become a third runtime.



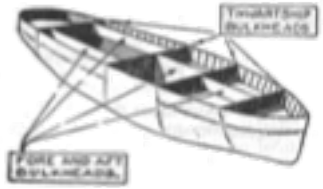
The Birth of CRI

The Kubelet should not vendor a runtime implementation.

(Unfortunately, this is not the original words literally)

- Developers from Google, CoreOS, and Hyper.sh drafted a kubelet-runtime interface together.
- The interface, CRI, was written with gRPC
  - gRPC had already been open sourced at that time.
  - The performance difference between gRPC and HTTP/REST was tested
- First CRI implementation: dockershim
- First Non-Docker CRI implementation: Frakti

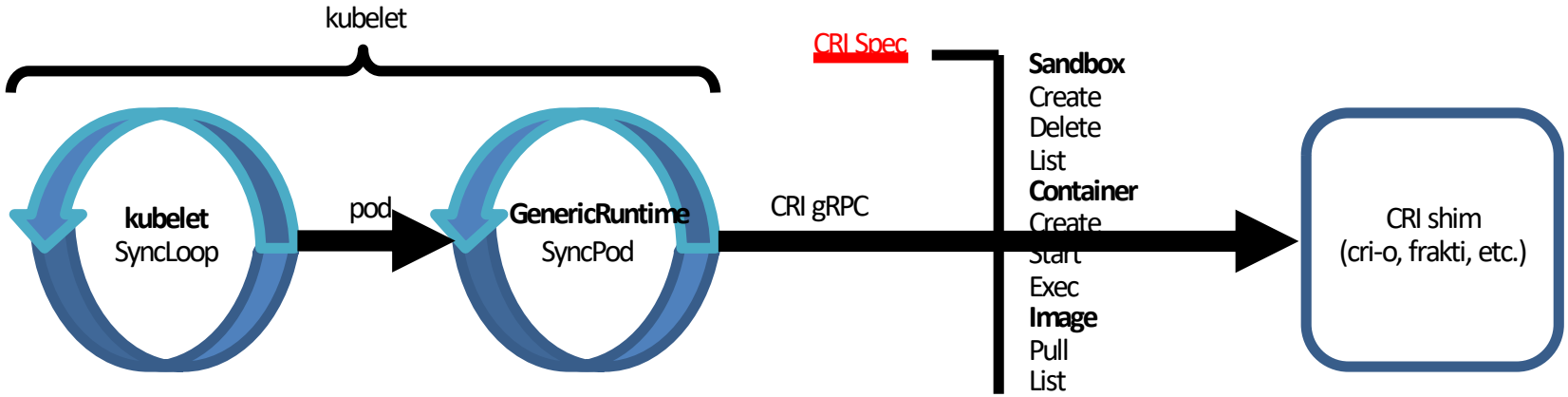
About Frakti



Picture from wikipedia

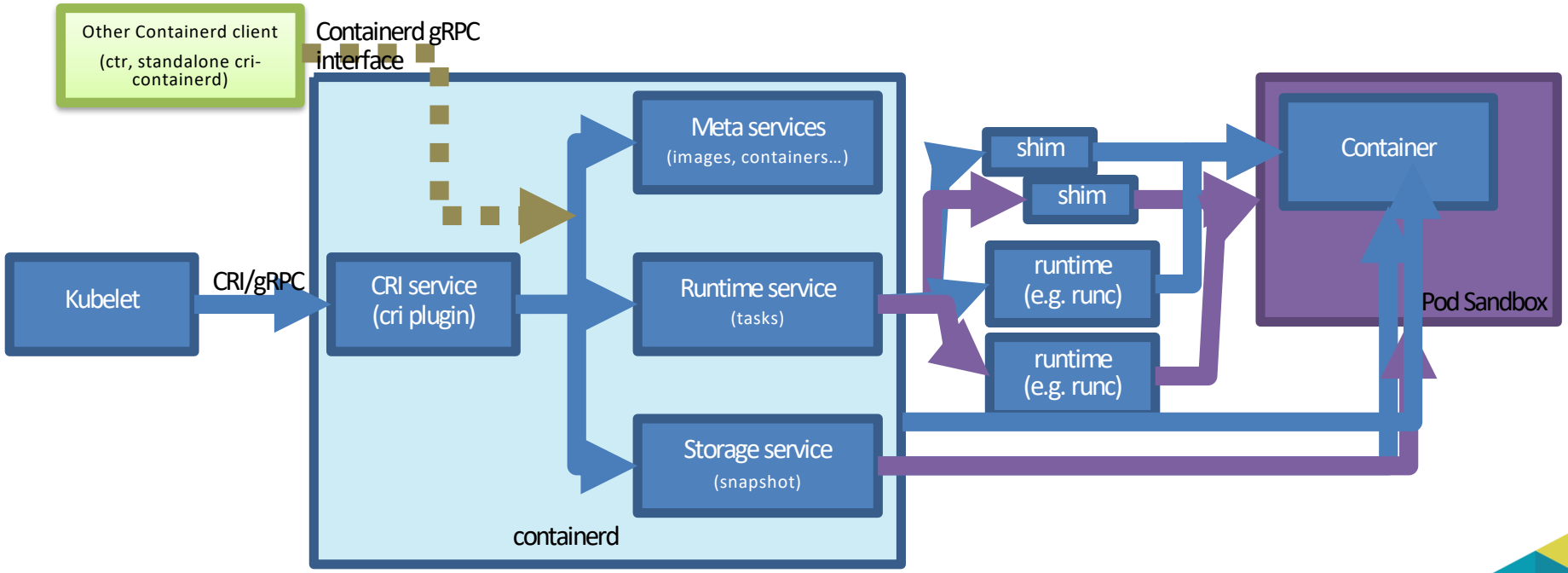
- Named with Greek word φρακτή
  - bulkhead, partition in ships, which keep ship safe by isolated the partitions of a ship with each other.
  - First seen in a book from Zhu Yu (朱彧), Song Dynasty, 12th century
  - *PS: Kubernetes () comes from Greek word Κυβερνήτης*
- CRI runtime Contributed by hyper.sh engineers
  - Run non-privileged containers with hyperd/runV
  - Run privileged containers with dockershim
  - And Unikernel (GSoC 2017, by ZJU)

The CRI Interface

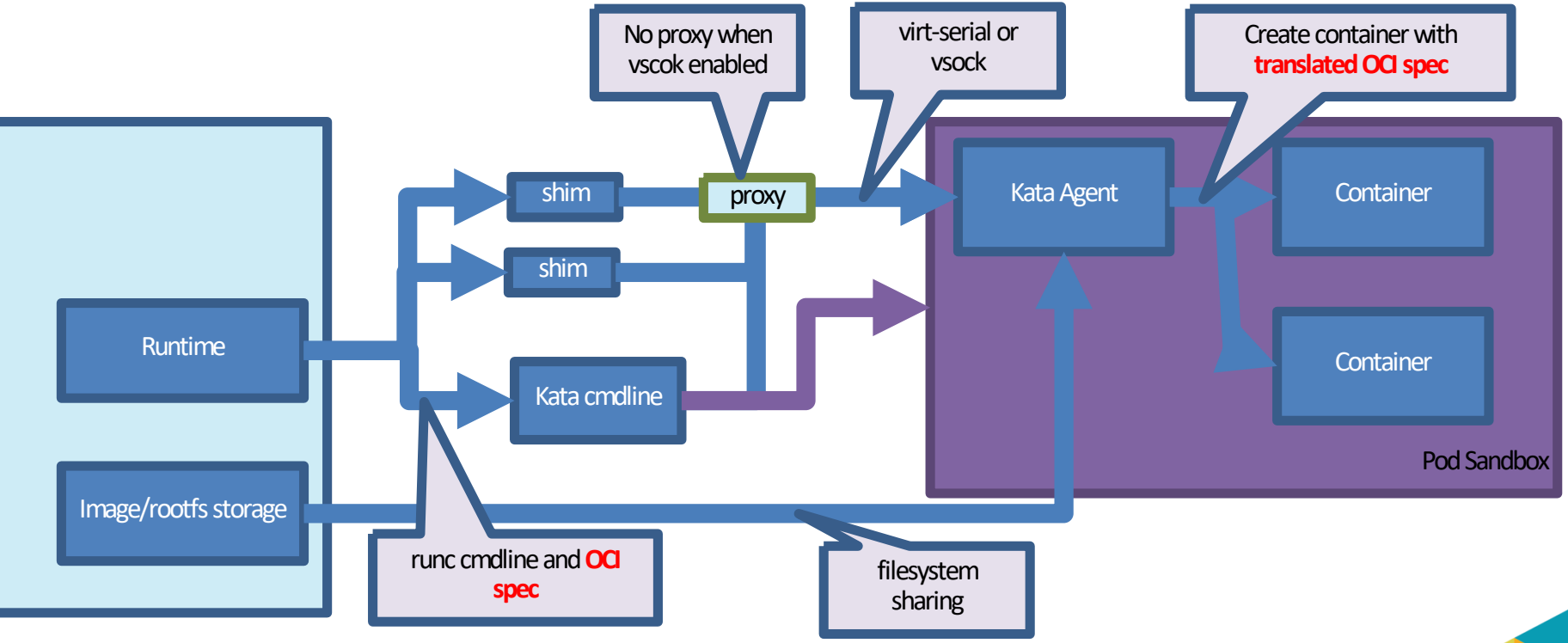


- Describe what kubelet expects from container runtimes
- Imperative container-centric interface
- Extensibility

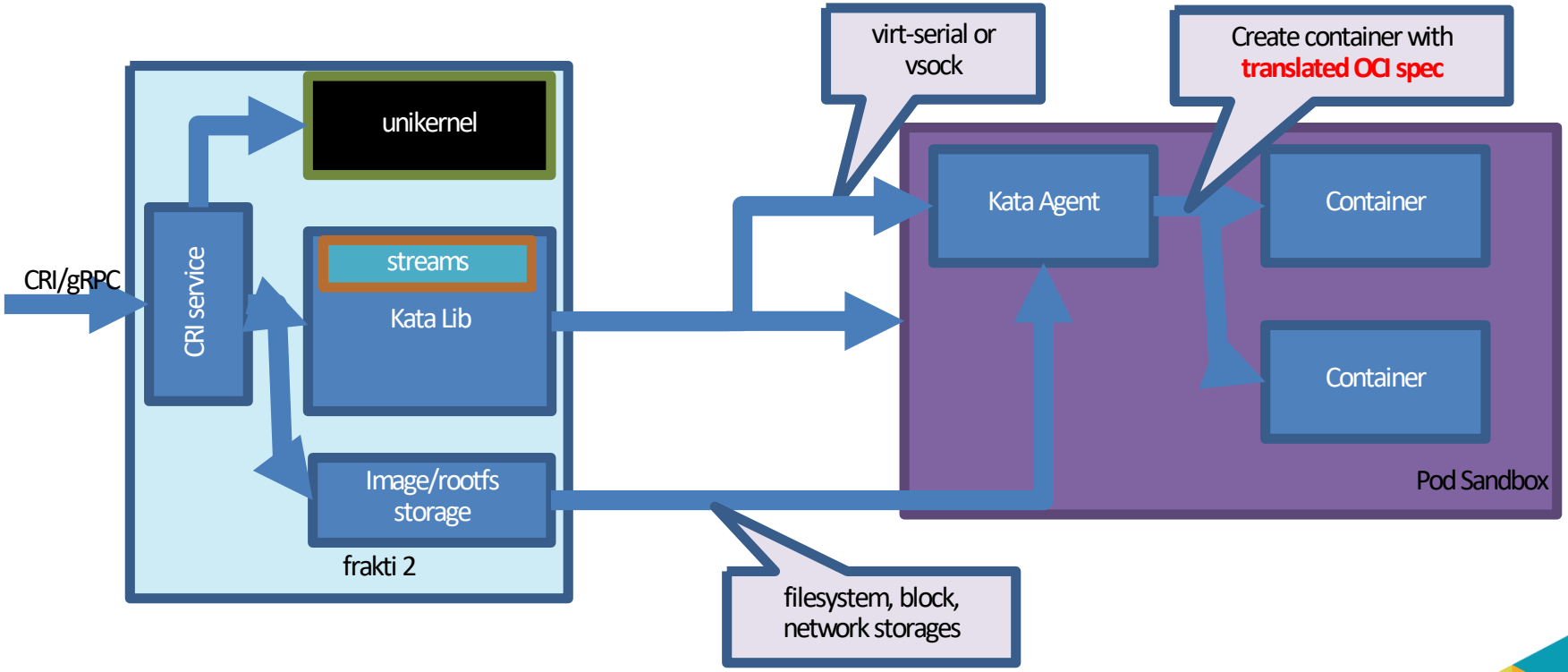
Containerd (w/ cri plugin) in brief



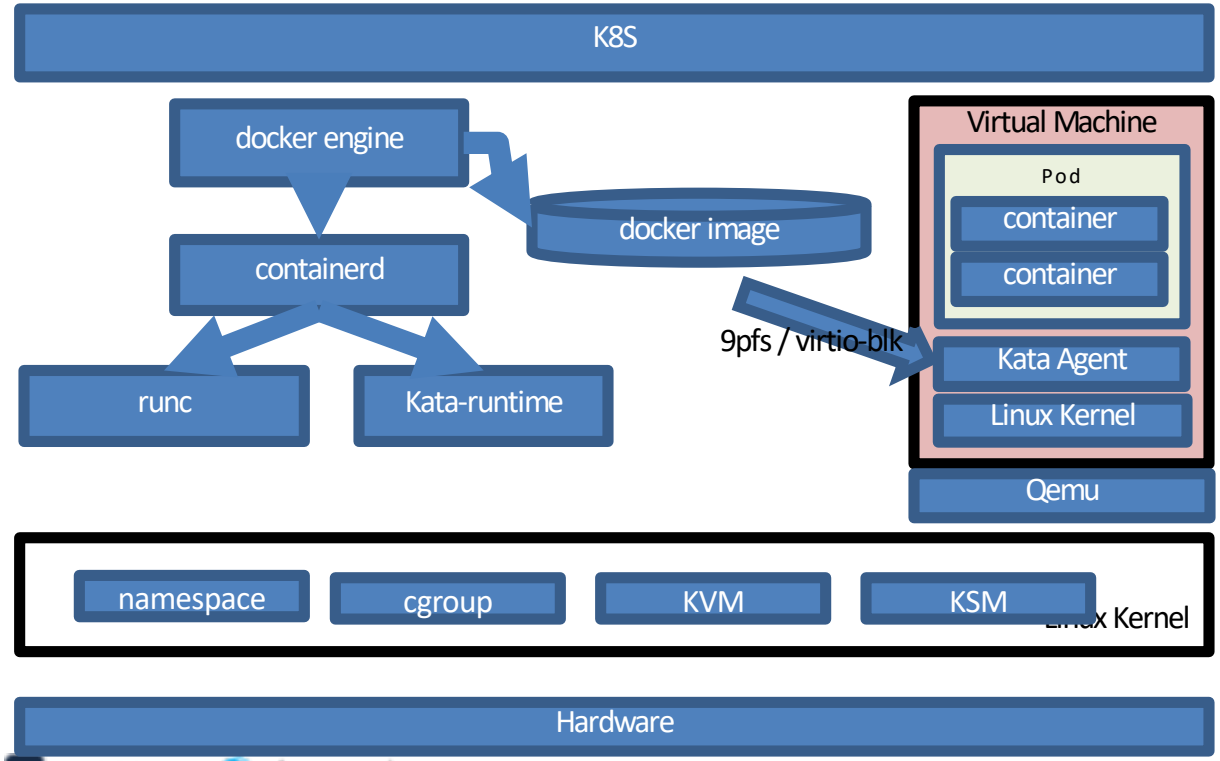
containerd/CRI-O + Kata



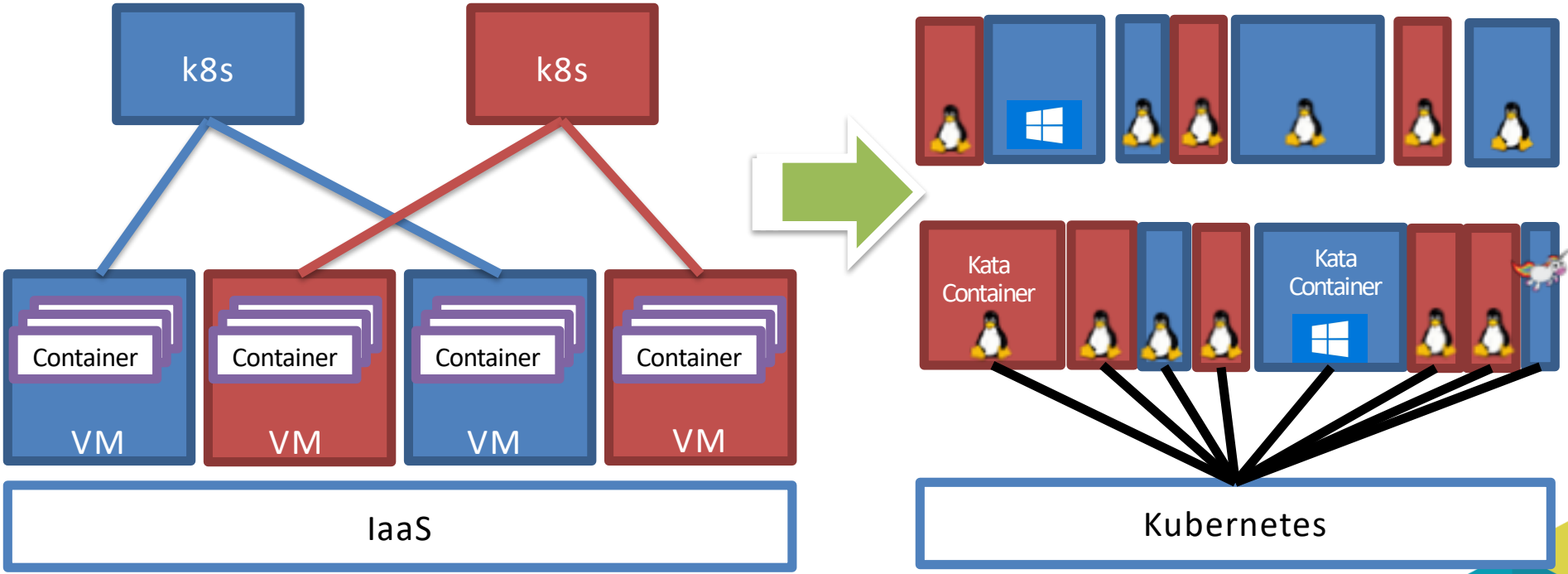
Frakti + Kata


virt-serial or vsock
Create container with translated OCI spec
unikernel
CRI/gRPC
CRI service
streams
Kata Lib
Image/rootfs storage
frakti 2
Kata Agent
Container
Container
Pod Sandbox
filesystem, block, network storages

Docker+kata


K8S
docker engine
containerd
runc
Kata-runtime
docker image
9pfs / virtio-blk
Virtual Machine
Pod
container
container
Kata Agent
Linux Kernel
Qemu
namespace
cgroup
KVM
KSM
Linux Kernel
Hardware

Kata: Remove an Indirect Layer



Get Started

JOIN THE COMMUNITY

Components overview

1. Github organization： https://github.com/kata-containers/
2. Runtime: OCI spec compatible and runc cmdline compatible runtime, and provides virtcontainers Libs for embedded use for Frakti
3. Agent: run inside VM as init or systemd service, wrapper of libcontainer.
4. Shim： one per container process, used for I/O transfer
5. Proxy： proxy the CMD and I/O to kata-agent via serial
6. Linux: 4.14， minimum workable Guest Kernel
7. Qemu: 2.11， optimized hypervisor with better performance and smaller memory footprint
8. Others: osbuilder, ci, tests, packaging, community, documentation, ksm-throttler

Contribute

- Website: https://katacontainers.io
- Code and documentation hosted on https://github.com/kata-containers/
- Major releases managed through Github* Projects
- Intel (Intel® Clear Containers) & Hyper.sh (runV) contributing initial IP
- Apache 2 license
- Slack: katacontainers.slack.com
- IRC: #kata-dev@freenode
- Mailing-list: kata-dev@lists.katacontainers.io

Open Governance

- **Contributors**
  - At least one github contribution for the past 12 months
- **Maintainers**
  - Active contributor, nominated by fellow maintainers
  - Can merge code
- **Architecture Committee**
  - Take high level architecture and roadmap decisions
  - 5 seats, elected by contributors
  - Initial Members: Hyper.sh, Intel, Huawei, Google, Microsoft

Kata Containers in One Page

- Kata Containers is a container runtime like runC, but
  - Implemented with virtualization technology
  - Secured, and support other accelerating technologies for VMs
- Though virtualization based, Kata is lightweight and fast
- And Kata Containers support multiple Architecture:
  - x86_64, arm64, ppc64le etc.
- For Kubernetes, Kata enables
  - Runtime layer hard multi-tenancy
  - Heterogeneous OS container in a same pool

Thank You